Amendments to the Specification:

In Section [0005]

Fig.1 is a diagram of the~~an~~ internal data memory 12 in the MCS-51 series processor according to the prior art. The ~~data~~ memory 12 is shared for use as a stack memory, a data memory, and a register memory. Addresses for the stack, data, and registers are all 8 bits, and the processor processes an 8-bit command set. ~~When~~ While accessing the internal data memory 12, 8-bit instructions and addresses are sent to a memory address generator 14 to generate memory addresses. The internal data memory 12 in the MCS-51 series processor is 128 bytes, while the internal memory in the MCS-52 series processor is 256 bytes. ~~An external data memory in the MCS-51/52 series processor can be extended to 64K bytes.~~ The internal data memory 12 is divided into several segments: a) addresses 00H-1FH: a 32-byte register bank comprising four working register banks which have 8 registers each, wherein data can be accessed by direct addressing or indirect addressing; b) addresses 20H-2FH: a bit-addressing segment of 16 bytes (128 bits), wherein bits in the bit-addressing segment can be operated on via bitwise operational instructions; c) addresses 30H-7FH: a general segment for use by the user, wherein a stack is usually located by appointing stack pointers to this segment; and d) addresses 80H-FFH: a general segment existing only in the MCS-52 series processors and accessible only by indirect addressing.

In Section [0006]

Fig.2 is a diagram of the internal data memory 12 illustrating the~~comprising~~ two stacks according to the prior art. The configuration of the internal data memory 12 is shown as in Fig.2. Some of the memory is used for registers and data, and the other memory is used for the stack. As described above, the stack is usually located in a~~the~~ general segment of the internal data memory 12, and the stack pointer points to a position of the general segment as a stack starting address in the beginning. The stack is used to store the program counter when calling subroutines or to store other data as specified by the user. As shown in Fig.2, the data stored in the stack is represented by the shaded area below the stack pointer, and the stack pointer moves upwards in a direction shown by the arrow when new data is pushed into the stack. However, when calling subroutines, many programs need to send parameters and use a software stack memory to store these parameters. The parameters stored in the software stack memory are shown as the shaded area above the software stack pointer in Fig.2, and the software stack pointer moves downwards in a direction shown by the arrow when new parameters are pushed into the software stack memory. Because the two stacks share the limited memory, the user has to be aware of how many memory resources the two stacks use when programming.

In Section [0007]

As described above, the conventional MCS-51/52 series processors provides a limited internal data memory 12. The stack memory, data memory, and register memory have to share the internal data memory 12. As the complexity of computer systems increase ~~increases~~, more stack memory and data memory is needed. ~~Although the internal data memory 12 can be extended through the use of an external data memory, the amount of stack memory is still limited by the internal data memory 12.~~ Accordingly, the stack memory is not adequate for complicated programs that need to call many subroutines. Limited stack memory requires programs to call only a limited number of subroutines. Moreover, if subroutines have to send parameters, more stack memory is needed. Because the MCS-51/52 series processors ~~process~~ use an 8-bit command set, ~~each of the memory addresses is 1-byte (8-bit) and~~ the internal data memory is limited to 256-byte. Therefore, when stack memory cannot be extended, a user has to be aware of the stack size and has difficulties in programming.

In Section [0011]

Fig.1 is a schematic diagram of an ~~interior data~~ internal memory of~~in~~ a conventional MCS-51 series processor.

In Section [0012]

Fig.2 is a schematic diagram of a conventional ~~interior data~~internal memory comprising two stacks.

In Section [0013]

Fig.3 is a schematic diagram of the~~an~~ ~~interior~~internal memory of the processor according to the present invention.

In Section [0014]

Fig.4 is a schematic diagram of a hardware stack and a software stack disposed in a stack memory ~~which is~~as depicted in Fig.3.

In Section [0015]

Please refer to Fig.3. Fig.3 is a schematic diagram of ~~an interior~~the internal memory of the processor 20 according to an embodiment of the present invention~~ processor 20~~. The ~~interior~~ internal memory of the processor 20 comprises a data memory 22 and a stack memory 24. A central processing unit (CPU) 26 in the processor 20 processes an 8-bit instruction set. ~~When not~~Without changing the instruction set, the largest ~~ranges of the CPU 26 for accessing the data memory 22 and the stack memory 24 are both 256 bytes~~size of stack memory is now 256 bytes, and the largest size of data memory is also 256 bytes. The data memory 22 is used to store data for executing programs, and functioning as registers when the CPU 26 operates. The method of accessing data memory 22 is substantially ~~the same as~~similar to conventional methods, with the ~~except~~exception that the data memory 22 does not store stack data.~~;~~ Therefore~~therefore~~, when the CPU 26 needs stack space to execute programs, stack data is stored in the stack memory 24, which is only used to store stack data. The CPU 26 uses the stack memory 24 referencing a stack pointer generator 30.

In Section [0016]

Please refer to Fig.4. Fig.4 is a schematic diagram of two stack pointers~~s~~ disposed in the ~~present invention~~ stack memory 24 of the present invention. Generally speaking, the hardware stack utilizes the stack pointer, which ~~starts~~ increases incrementally from a predetermined starting position to point to a next address~~, while~~. Meanwhile the software stack utilizes a software pointer that~~which~~ decreases from a predetermined starting position. For maximizing~~ the stack~~ usage~~, a better disposition~~ of the stack memory 24, ~~is~~

~~to set~~ the stack pointer is initially set to 00H as a start address of the hardware stack, and ~~to set~~ the software stack pointer is initially set to FFH as a start address of the software stack. The stack pointer of the hardware stack ~~starts~~ ~~incrementally~~ increments from the ~~lowest~~ start address of the stack memory 24 to point to a following~~next~~ address. The software stack pointer of the software stack decreases from the highest address of the stack memory 24. This is an example of the assignment of both stack pointers' starting address~~The stack pointer and the software stack pointer cannot be coincident, in other words, the largest amount of the two stacks cannot exceed the 256-byte stack memory 24. In addition, if the stack memory 24 is disposed with two stacks having the same size, the stack pointer is appointed to 80H as the start address of the hardware stack and the software stack pointer is appointed to 7FH as the start address of the software stack. Consequently, the stack pointer starts incrementally from 80H to the largest address FFH and the hardware stack is positioned in addresses 80H-FFH of the stack memory 24. The software stack pointer decreases from 7FH to the lowest address 00Hand the software stack is positionedin addresses 00H-7FH of the stack memory 24. The two stacks utilize half of the stack memory 24, namely 128 bytes.~~with the present invention.

In Section [0017]

Briefly summarized, th~~e present invention~~ processor 20 of the present invention comprises ~~the~~ a CPU 26, ~~the~~ a data memory 22, ~~the~~ a stack memory 24, ~~the~~ a memory address generator 28, and ~~the~~ a stack pointer generator 30. When executing programs, the CPU 26 accesses the data memory 22 through the memory address generator 28 so as to read data for executing programs and write data into the data memory 22. However, when executing stack related instructions~~ related to stacks~~, the CPU 26 accesses the stack memory 24 through the stack pointer generator 30. Therefore, the stack memory 24 is used for stack data only and the data memory 22 is shared by non-stack data.

In Section [0018]

Compared with the prior art, the present invention provides the processor with an unshared stack memory, while ~~the~~a conventional processor ~~just~~ has ~~a limited~~only one internal ~~interior data~~ memory shared by stack memory, a data memory, and register memory. ~~Although data memory can be extended through an exteriordata memory, the amount of the stack memory is still limited from the interior data memory provided by the CPU. This can bring about insufficient stack memory, especially when complex programs require multiple levels of subroutines. The present invention processor has a data memory and a stack memory, and thereby enhance the system.~~